

Docket No. AUS920010871US1

**APPARATUS AND METHOD OF USING XML DOCUMENTS TO PERFORM
NETWORK PROTOCOL SIMULATION**

CROSS-REFERENCE TO RELATED APPLICATIONS

5

This application is related to co-pending US Patent
Application Serial No. _____ (IBM Docket No.
AUS920010868US1), entitled APPARATUS AND METHOD OF
GENERATING AN XML SCHEMA TO VALIDATE AN XML DOCUMENT USED TO
10 DESCRIBE NETWORK PROTOCOL PACKET EXCHANGES by the inventors
herein, filed on even date herewith and assigned to the
common assignee of this application.

This application is also related to co-pending US
15 Patent Application Serial No. _____ (IBM Docket No.
AUS920010869US1), entitled APPARATUS AND METHOD OF
DIAGNOSING NETWORK PROTOCOL ERRORS USING XML DOCUMENTS by
the inventors herein, filed on even date herewith and
assigned to the common assignee of this application.

20

This application is further related to co-pending US
Patent Application Serial No. _____ (IBM Docket No.
AUS920010870US1), entitled APPARATUS AND METHOD OF
GENERATING AN XML DOCUMENT TO REPRESENT NETWORK PROTOCOL
25 PACKET EXCHANGES by the inventors herein, filed on even date
herewith and assigned to the common assignee of this
application.

BACKGROUND OF THE INVENTION

1. Technical Field:

5 The present invention is directed to communications networks. More specifically, the present invention is directed to a method and apparatus for debugging network protocol errors using an XML document.

2. Description of Related Art:

10 Most network application programs exchange data using data packets. Typically, a packet has a specific structure that incorporates internal fields that clearly delineate the packets' different contents. Using this structural representation, a user may devise algorithms that may be
15 used to effectuate network simulation testing to debug network problems etc. The algorithms may be devised using a markup language. A markup language is a language that allows additional text or tags that are invisible to users to be inserted into a document. Thus, the tags are not part
20 of the content of the document but rather enhance the document. For example, the tags may be used to structure the document or to add hypertext capability to the document etc.

One of the markup languages that is particularly well
25 suited for this task is the eXtensible Markup Language or XML. XML is a language that is especially designed for Web documents. It allows designers to create their own customized tags, enabling definition, transmission, validation, and interpretation of data between applications
30 and between organizations.

[illegible]

- 3 -

SUMMARY OF THE INVENTION

5 The present invention provides a method, system and
apparatus for performing network protocol simulation using
XML document. Network data packets are used to generate an
XML document. After analyzing the document, parts of the
document are then changed to simulate changes to the data
packets.

10

11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

10 Fig. 1 is an exemplary block diagram illustrating a distributed data processing system according to the present invention.

Fig. 2 is an exemplary block diagram of a server apparatus according to the present invention.

15 Fig. 3 is an exemplary block diagram of a client apparatus according to the present invention.

Fig. 4 depicts a TCP/IP data packet.

Fig. 5 depicts a TCP header format.

Fig. 6 is a sample XML document.

20 Fig. 7 depicts added elements to the sample XML document in Fig. 6.

Fig. 8 depicts an XML document representing generic packet exchanges of a TCP/IP setup connection.

25 Fig. 9 is a flow chart of a program that may be used to generate an XML document of a generic TCP/IP setup connection.

Fig. 10 is a flow chart of a process that may be used to implement a parser to parse an XML document.

30 Fig. 11 depicts an XML schema for a generic TCP/IP setup connection.

Fig. 12 depicts an XML document representing packet exchanges for a generic TCP/IP close connection process.

Fig. 13 is a flow diagram of a program that may be used to generate an XML document for a generic a TCP/IP close connection process.

5 Fig. 14 is a flow diagram of a parser that may be used to notify a user whether a generic close setup connection was successful.

Fig. 15 depicts an XML schema for packet exchanges in a generic TCP/IP close setup connection.

10 Fig. 16 depicts packet exchanges for a TCP/IP login setup connection.

Fig. 17 an XML document of the TCP/IP login setup connection.

15 Fig. 18 is a high level output of a parser that has parsed a TCP/IP data transaction from establishing a connection to closing the connection.

Fig. 19 is a first example of an XML document representing a generic TCP/IP setup connection that has not been well formed.

20 Fig. 20 is a second example of an XML document representing a generic TCP/IP setup connection that has not been well formed.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

With reference now to the figures, Fig. 1 depicts a pictorial representation of a network of data processing systems in which the present invention may be implemented. Network data processing system 100 is a network of computers in which the present invention may be implemented. Network data processing system 100 contains a network 102, which is the medium used to provide communications links between various devices and computers connected together within network data processing system 100. Network 102 may include connections, such as wire, wireless communication links, or fiber optic cables.

In the depicted example, server 104 is connected to network 102 along with storage unit 106. In addition, clients 108, 110, and 112 are connected to network 102. These clients 108, 110, and 112 may be, for example, personal computers or network computers. In the depicted example, server 104 provides data, such as boot files, operating system images, and applications to clients 108, 110 and 112. Clients 108, 110 and 112 are clients to server 104. Network data processing system 100 may include additional servers, clients, and other devices not shown. In the depicted example, network data processing system 100 is the Internet with network 102 representing a worldwide collection of networks and gateways that use the TCP/IP suite of protocols to communicate with one another. At the heart of the Internet is a backbone of high-speed data communication lines between major nodes or host

computers, consisting of thousands of commercial, government, educational and other computer systems that route data and messages. Of course, network data processing system 100 also may be implemented as a number of different
5 types of networks, such as for example, an intranet, a local area network (LAN), or a wide area network (WAN). Fig. 1 is intended as an example, and not as an architectural limitation for the present invention.

Referring to Fig. 2, a block diagram of a data
10 processing system that may be implemented as a server, such as server 104 in Fig. 1, is depicted in accordance with a preferred embodiment of the present invention. Data processing system 200 may be a symmetric multiprocessor (SMP) system including a plurality of processors 202 and 204
15 connected to system bus 206. Alternatively, a single processor system may be employed. Also connected to system bus 206 is memory controller/cache 208, which provides an interface to local memory 209. I/O bus bridge 210 is connected to system bus 206 and provides an interface to I/O
20 bus 212. Memory controller/cache 208 and I/O bus bridge 210 may be integrated as depicted.

Peripheral component interconnect (PCI) bus bridge 214 connected to I/O bus 212 provides an interface to PCI local bus 216. A number of modems may be connected to PCI local
25 bus 216. Typical PCI bus implementations will support four PCI expansion slots or add-in connectors. Communications links to network computers 108, 110 and 112 in Fig. 1 may be provided through modem 218 and network adapter 220 connected to PCI local bus 216 through add-in boards.

Additional PCI bus bridges 222 and 224 provide interfaces for additional PCI local buses 226 and 228, from which additional modems or network adapters may be supported. In this manner, data processing system 200 allows connections
5 to multiple network computers. A memory-mapped graphics adapter 230 and hard disk 232 may also be connected to I/O bus 212 as depicted, either directly or indirectly.

Those of ordinary skill in the art will appreciate that the hardware depicted in Fig. 2 may vary. For example,
10 other peripheral devices, such as optical disk drives and the like, also may be used in addition to or in place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention.

15 The data processing system depicted in Fig. 2 may be, for example, an IBM e-Server pSeries system, a product of International Business Machines Corporation in Armonk, New York, running the Advanced Interactive Executive (AIX) operating system or LINUX operating system.

20 With reference now to Fig. 3, a block diagram illustrating a data processing system is depicted in which the present invention may be implemented. Data processing system 300 is an example of a client computer. Data processing system 300 employs a peripheral component
25 interconnect (PCI) local bus architecture. Although the depicted example employs a PCI bus, other bus architectures such as Accelerated Graphics Port (AGP) and Industry Standard Architecture (ISA) may be used. Processor 302 and main memory 304 are connected to PCI local bus 306 through
30 PCI bridge 308. PCI bridge 308 also may include an integrated memory controller and cache memory for processor 302. Additional connections to PCI local bus 306 may be

made through direct component interconnection or through add-in boards. In the depicted example, local area network (LAN) adapter 310, SCSI host bus adapter 312, and expansion bus interface 314 are connected to PCI local bus 306 by direct component connection. In contrast, audio adapter 316, graphics adapter 318, and audio/video adapter 319 are connected to PCI local bus 306 by add-in boards inserted into expansion slots. Expansion bus interface 314 provides a connection for a keyboard and mouse adapter 320, modem 322, and additional memory 324. Small computer system interface (SCSI) host bus adapter 312 provides a connection for hard disk drive 326, tape drive 328, and CD-ROM drive 330. Typical PCI local bus implementations will support three or four PCI expansion slots or add-in connectors.

An operating system runs on processor 302 and is used to coordinate and provide control of various components within data processing system 300 in Fig. 3. The operating system may be a commercially available operating system, such as Windows 2000, which is available from Microsoft Corporation. An object oriented programming system such as Java may run in conjunction with the operating system and provide calls to the operating system from Java programs or applications executing on data processing system 300. "Java" is a trademark of Sun Microsystems, Inc. Instructions for the operating system, the object-oriented operating system, and applications or programs are located on storage devices, such as hard disk drive 326, and may be loaded into main memory 304 for execution by processor 302.

Those of ordinary skill in the art will appreciate that the hardware in Fig. 3 may vary depending on the implementation. Other internal hardware or peripheral devices, such as flash ROM (or equivalent nonvolatile

memory) or optical disk drives and the like, may be used in addition to or in place of the hardware depicted in Fig. 3. Also, the processes of the present invention may be applied to a multiprocessor data processing system.

5 As another example, data processing system 300 may be a stand-alone system configured to be bootable without relying on some type of network communication interface, whether or not data processing system 300 comprises some type of network communication interface. As a further example, data
10 processing system 300 may be a Personal Digital Assistant (PDA) device, which is configured with ROM and/or flash ROM in order to provide non-volatile memory for storing operating system files and/or user-generated data.

15 The depicted example in Fig. 3 and above-described examples are not meant to imply architectural limitations. For example, data processing system 300 may also be a notebook computer or hand held computer in addition to taking the form of a PDA. Data processing system 300 also may be a kiosk or a Web appliance.

20 The present invention provides an apparatus and method of using XML documents to perform network protocol simulation. The invention may be local to client systems 108, 110 and 112 of Fig. 1 or to the server 104 or to both the server 104 and clients 108, 110 and 112. Consequently,
25 the present invention may reside on any data storage medium (i.e., floppy disk, compact disk, hard disk, ROM, RAM, etc.) used by a computer system.

30 The bulk of communications occurring over the Internet is done using TCP/IP (Transmission Control Protocol/Internet Protocol). Accordingly, the present invention will be described using TCP/IP. Nonetheless, it should be understood that the invention is not restricted to only

TCP/IP. Any other type of network communication protocol may be used and would be well within the scope and spirit of the invention.

5 **OVERVIEW OF INTERNET COMMUNICATIONS**

Since TCP/IP will be used to explain the present invention, a general description of TCP/IP is therefore warranted. The TCP/IP protocol is typically implemented as a layered protocol stack where data packets are processed layer by layer. As an example, a typical network transaction using TCP/IP is the transfer of e-mail messages over the Internet. For a user to send an e-mail message to a recipient, the user has to fill in the e-mail address of the recipient and type in the text of the message. Then, the user has to assert the "send" button.

When the "send" button is asserted, the text of the message (or the message) is sent to a TCP layer. If the message is too long, for example when a large file is attached to the message, the TCP layer will break the message up into datagrams or data packets and adds a header in front of each data packet. The TCP header will be described later. The TCP layer will then send each data packet (including the added header) to an IP layer. The IP layer then puts an IP header to the data packet that includes a source IP address and a destination IP address. Using the IP addresses, each data packet will then be sent to the recipient over the Internet.

Fig. 4 depicts each data packet that is transmitted over the Internet. As stated above, TCP header 405 is first added to user data 410 (e.g., data packet). Then, IP header 400 is added. Once this is completed, the data packet is allowed to enter the Internet. The IP header ensures that

the data reaches the target computer system while at the same time it lets the target system know where the message originates. In the case of accessing Web pages, the IP application protocol may be regarded as the application
5 program that opens up a communication line between the two computer systems so that data may be transmitted back and forth.

Upon receiving a data packet, the target computer system sends the packet to an IP layer where the IP header
10 is stripped off. The resulting data packet is then sent to a TCP layer. The TCP layer then strips the TCP header off the packet and collects all the packets in order to reconstruct the message. Once reconstructed, the message is sent to a mail application protocol. Using the e-mail
15 address of the intended recipient, the mail application protocol then puts the message into the mailbox of the recipient.

TCP HEADER

20 Since the IP header is not important to explain the invention, it will not be described. The TCP header will now be briefly described. Fig. 5 depicts a TCP header format. The first two bytes of the TCP header is 16-bit source port number 500. The next two bytes of the TCP
25 header is the 16-bit destination port number 505. The port numbers are used to keep track of different conversations. For example, if a server is communicating with three different clients, the server will use a particular port number to communicate with each one of the clients. Thus,
30 the 16-bit source port number 500 and the 16-bit destination port number 505 in conjunction with the IP address in the IP

header identify a unique connection. This unique connection is often referred to as a socket.

Each datagram or data packet has a 32-bit sequence number 510. The sequence number is used to let the receiving computer system know the order of the particular packet in the stream of packets. It is also used by the receiving computer system to notify the sending computer system that all packets have been received up to a certain number. TCP does not number the datagrams but rather numbers the octets (8-bit data) in each datagram. Thus, if there are 500 octets in each datagram or packet, the first datagram may have a sequence number of "0", the second "500", the third "1000" etc.

In order to ensure that a datagram has been received, the recipient has to send back a 32-bit acknowledgement response to the sender. For example, if a recipient sends an acknowledgement of 1500, it is telling the sender that it has received all the data up to octet number 1500. If the sender does not get an acknowledgement response within a pre-determined time, it will resend the data. When a data sender receives a new value, it can dispose of data that was held for possible re-transmission. The acknowledgement number is only valid when ACK flag 530 is set.

The 16-bit window size 555 represents the number of bytes starting with the byte specified in the acknowledgement number field 510 that the receiver is willing to accept. Stating differently, the window is used to control how much data can be in transit at any one time. It, in a way, advertises the amount of buffer space that has been allocated for the connection. The window size is used because it is not practical to wait for each datagram to be acknowledged before sending the next one, lest data

transactions over the Internet may be too slow. On the other hand, a sender cannot just keep sending data, or a fast computer system might overrun the capacity of a slow one. Thus, each computer system indicates how much new data it is currently prepared to absorb by putting the number of octets in its 16-bit window. As a recipient receives data, its window size will decrease until it reaches zero (0). At that point, the sender has to stop. As the receiver processes the data, it will increase its window size signaling that it is able to accept more data. Often times, the same datagram may be used both to acknowledge receipt of a set of data and to give transmission permission for additional new data.

The 4-bit header length 520 indicates the size of the entire TCP header. In Fig. 5, options, padding, reserve and a few other fields are not shown. The options field depends on the number of options set and thus is of variable length. Accordingly, there is not a pre-determined length for the TCP header. Hence, the length of each header has to be indicated.

When one-bit URG 525 is used, it indicates that the 32-bit urgent pointer field 565 is valid. As mentioned before, when one-bit ACK 530 is set, the 32-bit acknowledgement number 515 is valid. One-bit PSH 535 is used to instruct the receiver to pass the data received thus far immediately to the receiving application. RST 540 is used to tell the receiver to re-establish connection. This usually indicates that an error condition has been detected. SYN bit 545 synchronizes the sequence numbers to begin a connection and FIN bit 550 indicates that the sender has sent all data in a stream. If both ends of a communication have sent the FIN flag, the connection will be closed.

The 16-bit checksum 560 ensures that the TCP header and data have not been modified in transit. If the checksum is invalid, the receiver will not acknowledge the message. The value in 16-bit urgent pointer 565 points to the end of data field that is considered urgent and requires immediate attention. This field is not valid if URG bit 525 is not set.

ESTABLISHING A TCP/IP CONNECTION

10 To establish a TCP connection, an active computer system (e.g., a client) has to initiate communication with a passive computer system (e.g., a server) by sending a SYN packet (i.e., a packet with SYN bit 545 set) with the sequence number 510 set to an arbitrary value J. The server
15 will then respond with a SYN, ACK packet (i.e., both the SYN bit 545 and the ACK bit 530 are set) with the acknowledgement number 515 set to J+1 and the sequence number 510 set to a further arbitrary number K. The client then responds to the SYN, ACK packet with an ACK packet with
20 the acknowledgement number set to K+1. Note that in this case, both K and J are integers. Note also that only the parameters of importance for the connection to be established are described. However, other parameters such as window size etc. will also be included in the packets.
25 Once the connection is established, user data packets may then be transmitted.

The above scenario may be interpreted as the client and server negotiating parameters such as window size etc. to use when transferring the user data packets. The smaller of
30 the two parameters are used to actually transmit the user data.

CLOSING A TCP/IP CONNECTION

The TCP/IP connection may be closed when the application program running on the client makes a close () system call on the open socket. When this occurs the client
5 will send a FIN packet (i.e., the FIN bit 550 set) to the server with the sequence number 510 set to J. When the server receives the FIN packet, it passes an "end-of-file" indication to the software. At that time, the server will send an ACK packet to the client with the acknowledgement
10 number 515 set to J+1. The server will again send another packet, a FIN packet to the client with the sequence number set to K. The client will then respond with an ACK packet with a K+1 acknowledgement number. The TCP connection will then be closed.

15 Note that there are many existing methods of closing a TCP/IP connection. The method outlined above is the most often used method.

BRIEF DESCRIPTION OF AN XML DOCUMENT

20 Fig. 6 is an example of an XML document. The header of the document tells a user that this is an XML document that has been written using version 1.0 of the XML specification. The greater than (">") and the less than ("<") signs are tags. They indicate the opening and closing
25 of an element. Elements are the basic building blocks of an XML document. They may contain text, comments, or other elements. Every opening element (i.e. "<company>") must also contain a closing element (i.e. "</company>"). The closing element consists of the name of the opening element,
30 prefixed with a slash ("/").

XML is case-sensitive. While "<company ></company>" is well-formed, "<COMPANY></company >" and "<Company></COMPANY

>" are not. Also, if the element does not contain text or other elements, the closing tag may be abbreviated by simply adding a slash ("/") before the closing bracket in the element (e.g., "<company></company>" can be abbreviated as
5 "<company />"). In addition to the rules defining opening and closing tags, it is important to note that in order to create a well-formed XML document, the elements must be properly nested.

All attribute values must be contained within quotation
10 marks. For example, id="1" is correct, while id=1 is not acceptable. Where elements represent the nouns contained in an XML document, attributes represent the adjectives that describe the elements.

Thus in the XML of Fig. 6, a company and two of its
15 employees are defined. The relationship between the company (parent) and the employees (children) are also described. Note that new employees can easily be added. Fig. 7 depicts elements that are added to the example of Fig. 6.

In summary, XML is a text-based meta-language that uses
20 tags, elements, and attributes to add structure and definition to documents. It is a markup language because it uses tags to mark-up documents and it is a meta-language because it uses the tags to give structure to documents that is in turn used as a means of communication. XML is
25 extensible because it enables users to create their own collection of tags.

GENERATING AN XML DOCUMENT TO REPRESENT TCP/IP DATA TRANSACTIONS

30 Knowing the connection establishment, the transition state of each user data packet and the close connection procedures of TCP as well as the rules required to implement

an XML document, a software program may be written to convert TCP data transactions into an XML document. The document may then be sent to an XML parser to investigate network communications problems. Both the software program and the parser may be written in C, C++, Java or any other suitable programming language. The TCP/IP transactions may be acquired through an existing application program such as TCPdump, IPtrace, IPreport etc. or through a network sniffer. A network sniffer is a program or device that monitors data traveling over a network communications line.

Fig. 8 depicts an XML document representing a generic TCP/IP connection setup. As mentioned earlier, the TCP/IP connection setup uses three data packets, each packet of course contains an IP header and a TCP header. In the example of the TCP/IP connection above, the IP header and the TCP header are taken into consideration only once. Nonetheless, the IP header and TCP header of each packet are thoroughly examined for relevant information. For example, all invariant header attributes such as port numbers and IP addresses may be captured as attributes of the header tag. In any case, the IP_header is a parent element that contains a child element "TCP_header". The "TCP_header" element in turn contains child element "TCP_connection" and the "TCP_connection" contains children elements "SYN_sent", "SYN_received", "ACK_received" and "ACK_sent".

Fig. 9 is a flow chart of a program that may be used to generate the XML document of the TCP/IP connection setup described above. This flow chart assumes that all the data packets have an IP header and a TCP header. Of course, a program may be written to determine that it is indeed so. In any case, assuming that there are both an IP header and a TCP header, the present program will ensure that an IP

header element and a TCP header element are opened and closed in accordance with the above example. Note that here, only the first three packets are taken into consideration since per TCP/IP specification the first three
5 packets in any TCP/IP transactions are used to establish a TCP/IP connection.

The process starts when the program begins to execute (step 900). When the program gets the first packet, it determines whether the SYN flag bit 545 is set. If it is
10 not set, the program will go on looking at the next packet in the stream of packets to determine if the SYN bit is set in that packet (steps 902 and 904). The first packet may not have the SYN bit set if, for instance, it is not part of the TCP/IP transactions being investigated. To ensure that
15 the packet is part of the TCP/IP transactions being investigated the program may take into consideration the IP addresses in the IP header as well as the port numbers in the TCP header.

Note that the two IP addresses and the two port numbers
20 will alternate based on the computer system that sends the data packet. For example, when the client sends a packet, its IP address will be the source IP address and the IP address of the server will be the destination IP address. If, on the other hand, the server sends the packet, its IP
25 address will be the source IP address and the IP address of the client will be the destination IP address. Likewise, when the client sends the packet the port number that it is using for the connection will be the source port number and the port number that the server is using for that particular
30 connection will be the destination port number. The source and destination port numbers will be reversed when the server sends the packet.

After ensuring that the packet is the first one in the transactions and the SYN bit is not set then the program will not open and close the SYN_sent element in the XML document being generated. If the SYN bit is set, the SYN_sent element will be opened and closed (steps 902 and 906). Next a check will be made to determine whether there is a sequence number in the packet. If so, the number will be inserted between the opened and closed SYN_sent element. If not, a number will not be inserted (steps 908, 910 and 912). The next packet will then be investigated to determine whether both the SYN flag and the Ack flag are set. If so, a SYN_received and an ACK_received element will be opened and closed. Next, checks will be made as to whether there are a sequence number and an acknowledgement number. If so, the sequence number will be inserted between the opened and the closed SYN_received element and the acknowledgement number between the opened and closed ACK_received element (steps 916, 918, 920, 922, 924, 926, 928, 930 and 932).

The next packet will be checked to see whether the ack flag is set. If so, the ACK_sent element will be opened and closed and the acknowledgement number will be inserted between the opening and the closing tags of the ACK_sent element if one exists (steps 936, 938, 940, 942, 944 and 946). The execution of the program then ends (step 948).

A parser may be implemented to notify a user as to whether the TCP/IP connection sequence was proper. Fig. 10 is a process that may be used to implement the parser. In this case, the XML document generated above will be fed into the parser. The process starts with the execution of the parser (step 1000). The parser will check to see whether there are a SYN_sent element and a sequence number between

the opened and closed SYN_sent element. If not, an appropriate error message may be generated (steps 1002, 1004, 1006 and 1008). Then the parser will check to determine whether there are a SYN_received element and a number between the opened and closed SYN_received element. If not, an appropriate error message may be generated (steps 1010, 1012, 1014 and 1016). The parser will continue to check to see whether there are an ACK_received element and an ACK_sent element, whether there is a number between the opened and closed ACK_received and ACK_sent elements and whether these two numbers are the expected numbers. If not, appropriate messages may be generated; otherwise, a "connection setup successful" message may be generated (steps 1018 - 1042).

For the application presenting the XML document to the user to properly interpret the markup tags, a schema must be developed. As alluded to before, the purpose of an XML schema is to define and describe a class of XML documents by using schema components to constrain and document the meaning, usage and relationships of the constituent parts of the documents. Schemas may also provide for the specification of additional document information, such as normalization and default attribute and element values. Schemas have facilities for self-documentation. Thus, an XML schema can be used to define, describe and catalogue XML vocabularies for classes of XML documents.

Fig. 11 depicts an XML schema for the generic TCP/IP setup connection. In the schema, IP_header, TCP_header, SYN_sent, SYN_received, ACK_received and ACK_sent are all defined as elements. Their types are also defined (e.g., complextype or simpletype). In this case, "ref" is used for

simpletype. Sequence is a compositor that defines an ordered sequence of sub-elements or children. Note that each element that is opened is also closed. Note also that the schema is developed based on the state transition of the packets being transmitted (i.e., SYN, SYN&ACK and ACK packets). Thus, a schema may be developed for any packet state transitions. Once a schema is developed, the entries in the XML document may correctly be interpreted.

Note that an XML document may be generated for all data packets including the packets used during the TCP/IP close connection sequence. As before, an XML schema must be developed to correctly interpret the elements.

Fig. 12 depicts an XML document representing a generic TCP/IP close connection sequence. As with the TCP/IP setup connection process, a program may be written to automatically generate the XML document of the close connection sequence. In this case, a check will be made to ensure that both ends of the TCP/IP connection have sent a FIN packet. If so, the program will ensure that the proper elements are opened and closed if they are present and numbers are inserted in the proper place if present just as was done in the TCP/IP connection setup. A parser may be generated to notify the user as to whether the close connection process was properly executed. If not, appropriate error messages will be generated. Otherwise, a "close connection setup successful" may be generated.

Fig. 13 is a flow diagram of a program that may be used to generate the XML document outlining the TCP/IP close connection setup. The program will check to ensure that both ends of the network transaction have sent a FIN packet as per the XML specification. If so, then the TCP/IP connection is being closed. Consequently, the program will

ensure that the four packets, starting with the first FIN packet, are the proper packets and the program will open and close a FIN_sent element, an ACK_received element, a FIN_received element and an ACK_sent element and the appropriate numbers will be inserted between each open and close element (steps 1300 - 1354).

Fig. 14 is a flow diagram of a parser that may be used to notify the user whether the close setup connection was successful. The parser will ensure that all the open and close elements are present and in the proper sequence in the XML document. The parser will also ensure that the proper numbers are inserted between an open and close element. If there is any discrepancy between what is expected and what is actually in the document, the parser may generate an error to notify the user (steps 1400 - 1440).

Again a schema needs to be generated to validate the XML document representing the close connection sequence. Fig. 15 is a schema for the close connection sequence.

The TCP/IP setup connection process in Fig. 8 was for a generic connection. Fig. 16 depicts a TCPdump for a TCP/IP packet exchange for a remote login connection setup. A TCPdump is publicly available program that captures and outputs the TCP packet exchanges between two end points of a network connection. Each line in Fig. 16 represents a packet. The first line (first packet) may be deciphered as TCP port 1023 on host "gil" sending a SYN packet to the login port on host "devo". The sequence number is 768512 and contained no data. The window size is set at 4096 and the maximum segment size is 1024. In the second line (second packet) host "devo" replied with a SYN, ACK packet. The sequence number is 947648 and it also contained no data. The acknowledgement number is 768513 which acknowledges the

afore-said SYN packet. The window size is 4096 and maximum segment size 1024. In the third line (i.e., third packet) "gil" responded with an ACK packet and the acknowledgement number is 947649 and window size is 4096. At that point the
5 connection is opened.

The XML document representing this specific TCP/IP connection setup is illustrated in Fig. 17. Here, attributes to the TCP_header are local and remote ports (i.e., 1023 and login), local and remote IP addresses (i.e.,
10 gil and devo) and the application initiating the TCP/IP setup connection (i.e., rlogin). Note that the IP addresses are expressed in terms of the names of the computer systems. It is well known in the field that if the name of a computer system is known, its IP address may easily be obtained.

15 In this case, the reverse address resolution protocol (RARP) may be used to find the IP address. ARP (address resolution protocol) is the protocol used by TCP/IP to convert a physical address into an IP address. A computer system wishing to find out an IP address of another computer
20 system broadcasts an ARP request onto the network. A computer system on the network that has the IP address responds with its physical address. RARP, on the other hand, is used to obtain a computer system's own IP address. A computer system wishing to find out its own IP address
25 broadcasts its own physical address on the network and the RARP server (the server that assigns IP addresses to the computer systems in the network) will reply with the computer system's IP address.

In any case, a program may be written to generate the
30 specific TCP/IP connection outlined above. Furthermore, a parser may be written to investigate any network communications problem that a user may encounter.

As with the TCP/IP setup connection, based on the state transition diagram of this specific TCP/IP connection, an XML schema may be developed for proper interpretation of the elements.

5 An XML document for user data may also be generated. This would include the TCP/IP setup connection, user data packet transactions and the close connection sequence. Of course, an XML schema will also have to be developed for proper interpretation of the elements used. When the
10 document is passed through an appropriate parser, if no errors are encountered, the parser may generate an output such as that depicted in Fig. 18. Note that this is a high level view of the output of the parser.

15 **DEBUGGING**

As mentioned in the discussion above, a parser may be developed to investigate communications errors. The parser uses as input the XML document representing the packets exchanges. If the XML document is well formed, then there
20 are not any network communications errors. If the document is not well formed, the parser will pinpoint the errors. Figs. 19 and 20 depict two XML documents. Based on the specification of the TCP/IP setup connection, both XML documents are not well formed. Therefore, the TCP/IP
25 connections would not have been established. In Fig. 19, the SYN_Received element comes before the SYN_Sent element. This indicates then that the packets were not exchanged in the order specified in the specification and thus the reason why the connection was not established. A parser (e.g.,
30 Fig. 7) should quickly point that out.

The second XML document is missing the SYN_Sent packet altogether. Again, the parser should point this fact as the reason the connection was not established. In addition, neither one of the two XML documents would be validated
5 against the connection setup schema described above as the elements do not follow the proper sequence in the schema.

Note also that the parser will ensure that the proper numbers are present. For example, when setting up and closing a TCP/IP connection, the ACK number sent should be
10 the sequence number received plus one. If this is not so, the parser will notify the user of the discrepancy.

Thus, when network data transactions are expressed using XML documents, investigations of network communications errors are greatly simplified. Indeed, a
15 user may merely look at the generated document (i.e., a parser need not be used) to uncover the errors.

SIMULATION

Furthermore, a user may use the XML documents to
20 perform network protocol simulation. Clearly, any change made to the XML document is in effect a change made to the packet exchanges. Consequently, using the XML documents a user may analyze the properties of the packets, modify as well as create new exchanges and study the effects of the
25 changes on the packets. Thus, performance modeling and analysis may easily be performed using XML documents.

By modifying the network protocol's state transition diagram, the user can cause subtle/major changes in network behavior, traffic pattern, response pattern, response time,
30 congestion etc. Through network behavior analysis the user can visualize and analyze the effects of the modification. This can be illustrated graphically, for example. XML is a

useful tool for such analysis and using the technique described here will lead to a simple mechanism for specification of protocol behavior and the corresponding simulation and analysis of the behavioral response pattern.

5 The description of the present invention has been presented for purposes of illustration and description, and is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The
10 embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use
15 contemplated.